Javacript alapismeretek

Mi az a JavaScript, mire alkalmazzuk?

A JavaScript egy olyan programozási nyelv, melyet eleinte direkt a HTML hiányosságainak pótlására fejlesztettek ki. Segítségével létrehozhatunk interaktív weboldalakat, de ma már akár alkalmazásokat is írhatunk benne.
A HTML és a CSS nyelvvel ellentétben itt mát nem leíró nyelvről, hanem tényleges programozási nyelvről beszélhetünk.
Mielőtt megismerkednénk a nyelv alapjaival, fontos egy különbséget megemlíteni. A JavaScript nem összekeverendő a Java nyelvvel. A kettőnek szinte semmi köze egymáshoz, viszont a névhasonlóság miatt, ezeket sokan keverik.

Használata, szintaxisa

A JavaScript kódok futtatására csak egy böngészőre van szükségünk, ugyanis hasonlóan a HTML és a CSS nyelvekhez, ezt is tudja bármelyik böngésző futtatni.
Szintén hasonlóan a CSS-hez, hogy a JS kódokat is kétféleképpen tudjuk a weboldalunkba ágyazni. Vagy közvetlen a dokumentumunkban helyezzük el a kódot, vagy külső fájlként hívjuk be.

HTML-ben elhelyezve a szintaxis az alábbi:

```
<html>
<head>
</head>
<body>
	<script>
		document.write("Helló világ");
	</script>
</body>
</html>
```

Fontos tudni, hogy az ily módon megadott JS kódokat a dokumentumban bárhol elhelyezhetjük. Akár a fejlécbe, de akár a dokumentum végére is. Ez általában a megvalósítani kívánt funkció rendeltetésétől függ.

Külső fájlként használva:

```
<html>
<head>
</head>
<body>
	<script src="scripts.js"></script>
</body>
</html>
```

Ebben az esetben az src=”” attribútummal hivatkozunk a fájlunkra, amiben elegendő magát a JS

kódot elhelyezni.

Bármelyik megoldást alkalmazzuk, a <script> taget mindig le kell zárni a </script>-tel!
A kódunk végrehajtódni pedig ott fog, ahol elhelyeztük, vagy meghívtuk azt.

A fenti példában látható document.write() paranccsal a „Helló világ” szöveget írattuk az oldalra. Az esetek többségében ezt a parancsot használjuk valamilyen kimenet, HTML dokumentumba való írására. Az írást itt nem szó szerint kell érteni, mivel a HTML fájlunk nem fog módosulni, csak a gépünk memóriájában, amit végül megjelenít a böngésző.
Tehát dinamikusan tudunk olyan tartalmakat létrehozni, amiknek a pontos értékét előre nem ismerjük. Ilyen például egy számológép, képgaléria, játékok, stb...

Egy nagyon egyszerű példa dinamikus tartalom létrehozására:

```
<script>
            document.write( 2 + 2 );
            document.write( "<br />");
            document.write( 2 * 5 );
</script>
```

Ebben az esetben a böngészőnkben egy 4 -est és egy 10-est fogunk látni, mivel az első utasítással a két számot összeadtuk, míg a másodikban megszoroztuk.

Egészítsük ki kicsit a példát, hogy rendezettebb és informatívabb legyen:

```
<script>
            document.write( "Példa matematikai műveletekre: <br />"
);
            document.write( "Összeadás (2+2): " + (2 + 2) );
            document.write( "<br />" );
            document.write( "Szorzás (2*2): " + (2 * 10) );
</script>
```

Nézd meg a kapcsolódó oktató videót!

A kódunk a következő kimenetelt fogja eredményezni:

```
Példa matematikai műveletekre:
Összeadás (2+2): 4
Szorzás (2*2): 20
```

Az első szembeötlő különbség, hogy itt néhol használunk idézőjelet, néhol nem.
Általában idézőjel közé akkor teszünk valamit, ha azzal a karaktersorral nem szeretnénk műveletet végezni, csak kiíratni, vagy hozzáfűzni valamihez. A példa első sorában is ezt látjuk. Egyszerűen kiíratjuk a szöveget, a végén egy
 taggel, hogy azt ezt követőek új sorba kerüljenek.
A kód második sora is hasonlóan folytatódik, viszont itt a végén elhelyezünk már egy műveletet is, az összeadást. Mivel a JS nyelven karakterláncokat összefűzni és számokat összeadni is a + jellel

lehet, így a műveletet zárójelek közé helyeztük el, hogy a program (a matematikai szabályoknak megfelelően) először a zárójelben lévőket dolgozza fel. Miután ezzel végzett és megkaptuk a 4-est, ezt az eredményt hozzáfűzi az *Összeadás (2+2):* karakterekhez, az azt követő + jel hatására. Harmadik sorban csak egy egyszerű
 elemet íratunk ki, majd a negyedikben ismét egy karakterláncot fűzünk egy művelet eredményével össze.

Ezzel a szisztémával bármennyi műveletet és szöveget összefűzhetünk, például:

```
document.write( "2 meg 2 az " + (2 + 2) + " Viszont 2 szorozva 2-vel szintén " + (2 * 2) + "  lesz." );
```

Eredménye:

```
2 meg 2 az 4 Viszont 2 szorozva 2-vel szintén 4 lesz.
```

Ennél összetettebbre igazából ritkán valósítjuk meg a kiírást, mert nagyon átláthatatlanná és nehezen módosíthatóvá teszi a programunkat. Bonyolultabb műveletekhez már változókat és függvényeket használunk, melyekkel szintén meg fogunk ismerkedni.
Viszont először lássuk azokat a nyelvi alapokat, amelyekre szükségünk lesz ezek használatához.

Nyelvi alapok

Változók deklarálása, kiíratása

Mik azok a változók?
Tulajdonképpen olyan azonosítóval ellátott értékek, amelyeket a programunkban feldolgozhatunk, módosíthatunk, műveleteket végezhetünk velük, vagy kiírathatunk.

Ha JavaScriptben szeretnénk a nevünket eltárolni későbbi felhasználásra, akkor azt az alábbi módon tehetjük meg:

```
var nevem = "Laci";
```

A var (variable = változó) szócskával adjuk meg a programnak, hogy most egy változót fogunk deklarálni.
Az ezt követő rész lesz a változó neve, amire a későbbiekben hivatkozunk.
Egy egyenlőség jelet követve pedig a változó értékét adjuk meg.

Ha szeretnénk a változó értékét kiíratni, akkor elég csak a nevére hivatkozni:

```
document.write( nevem );
```

Az előző részben látott karakterlánc összefűzésekét ebben az esetben is alkalmazhatjuk:

```
var vezeteknev = "Nagy ";
var keresztnev = "Laci";
```

```
document.write( vezeteknev + keresztnev );
```

Ugyanígy műveleteket is végezhetünk változókkal:

```
var elso       = 6;
var mmasodik = 5;

document.write( elso + masodik );
```

Eredményként természetesen 11-et fogunk kapni.

Változók deklarálásánál is nagyon fontos, hogy idézőjelekkel, van anélkül veszünk-e fel számot. Ha idézőjelek közé tesszük, akkor a program nem számként, hanem karakterláncként fogja kezelni az értékeket:

```
var elso       = "6";
var mmasodik = "5";

document.write( elso + masodik );
```

Ebben az esetben a két változó összefűzésre kerül, így kimenetül 65-öt fogunk kapni.

Az előző részben bemutatott utolsó példát, ha változókkal szeretnénk megoldani, akkor az alábbi felállást kell alkalmaznunk:

```
var elso      = 2;
var masodik   = 2;
var osszeadva = elso + masodik;
var szorozva  = elso * masodik;

document.write( "2 meg 2 az " + osszeadva + " Viszont 2 szorozva 2-
vel szintén " + szorozva + "  lesz." );
```

Mint látható a két számot és a műveleteket is elkülönítettük és a kiíratásba csak a változókra hivatkoztunk, amelyek az eredményt tárolják.
Programunk így már sokkal átláthatóbb, módosítása is egyszerűbb és a legnagyobb előnye, hogy a változók további műveletekben is felhasználhatóak maradnak.

Operátorok

Mik azok az operátorok?
Olyan jelek, amelyekkel műveleteket, vagy változók értékeinek összehasonlítását végezzük.
Az előző példákban már két operátorral is megismerkedtünk, amely az összeadás és szorzás volt.

Alapvetően két részre oszthatjuk az operátorokat:

Aritmetikai

+ : Összeadás, vagy összefűzés
- : Kivonás
* : Szorzás

/ : Osztás

Logikai

== : Egyenlő
!= : Nem egyenlő
! : Nem
< : Kisebb
> : Nagyobb
<= : Kisebb, vagy egyenlő
>= : Nagyobb, vagy egyenlő
|| : VAGY
&& : ÉS
true : Igaz
false : Hamis

A logikai operátorokat legtöbbször eldöntendő műveleteknél használjuk. Ezekre részletesen a következő részben térünk ki.

Feltételek

A feltételekkel lehetőségünk van a programunk más-más részeinek futtatására, attól függően, hogy mit kötünk ki feltételnek az egyes kódrészekhez. Feltételek írásakor használjuk a legtöbbször a logikai operátorokat, a továbbiakban ezekre láthatunk példákat.

Egy nagyon egyszerű példa feltételre:

```
if ( 2 == 2 )
{
	document.write( "Ez igaz!" );
}
```

Ebben a kis példában azt vizsgáljuk, hogy 2 egyenlő-e 2-vel. Feltételt deklarálni az if (ha) szócskával lehet, majd ezt követi zárójelek között maga a feltétel, jelen esetben a == (egyenlő) logikai operátorral. Végül a { } jelek között az a programrész helyezkedik el, amelynek le kell futnia a feltétel teljesülésekor.
Mivel 2 valóban egyenlő kettővel, így ki fog íródni az Ez igaz! Szöveg.

Ha azt kötöttük volna ki feltételnek, hogy 2 == 5, akkor a { } közötti kód soha nem futna le.

Lássunk ezzel a két számmal néhány példát, más operátorokkal:

```
if ( 2 != 5 ) {...}    - Ebben az esetben le fog futni a kódunk, mivel 2 nem egyenlő 5-tel.
if ( 2 < 5  ) {...}    - Szintén lefut a kódunk, hiszen 2 kisebb, mint 5.
if ( 2 > 5  ) {...}    - Nem fog lefutni a kódunk, hiszen 2 nem nagyobb, mint 5.
```

Egy feltételben több kritériumot is megadhatunk a Vagy és az ÉS operátorokkal:

```
if ( (2 == 2) && (3 == 3) ) {...}
```

A kódunk le fog futni, mivel azt vizsgáltuk, hogy 2 egyenlő-e kettővel, ÉS 3 egyenlő-e 3-mal.

Viszont ebben a formában már nem fog lefutni a kódunk:

```
if ( (2 == 2) && (3 == 5) ) {...}
```

Mivel az igaz, hogy 2 egyenlő 2-vel, de 3 már nem egyenlő 5-tel.
Ha a VAGY operátort használjuk, akkor már egészen más a helyzet:

```
if ( (2 == 2) || (3 == 5) ) {...}
```

Ebben az esetben lefut a kódunk, mert a két feltétel közül az első teljesülni fog. Tehát a feltételt ebben az esetben így értelmezzük:
Ha 2 egyenlő 2-vel, vagy 3 egyenlő 5- tel.

Feltételeket tetszőleges mennyiségű operátorokkal megírhatunk, semmi sem szab határt.
Természetesen konkrét adatok helyett, a legtöbbször változókat adunk meg a feltételekben is.

```
var szam1     = 10;
var szam2     = 15;
var osszeg    = szam1 + szam2;
var vizsgalt = 20;

if ( osszeg > vizsgalt )
{
        document.write(szam1 + " és " + szam2 " összege nagyobb mint " + vizsgalt);
}
```

Ebben azt vizsgáljuk, hogy a két összeadott szám nagyobb-e, mint 20. Mivel az eredmény 35 lesz, így megjelenik a kiíratás, melynek szövegébe a feldolgozandó számokat helyettesítettük be.

Feltételeknél nem csak azt adhatjuk meg, hogy mi történjen teljesülésnél, hanem azt is, hogy mi történjen amennyiben nem teljesül a feltételünk.
Az előző példa alapján:

```
if ( osszeg > vizsgalt )
{
        document.write( "Az eredmény nagyobb mint " + vizsgalt );
}
else
{
        document.write( "Az eredmény kisebb mint " + vizsgalt );
}
```

Az else (különben) ág akkor teljesül, ha maga a feltétel nem. Jelen példában akkor, ha az összeg nem nagyobb, mint a vizsgált szám.
Feltételeket szinte végtelen sorba is köthetünk, ezzel elég összetett elágazásokat megvalósítva.
Szintén az előző példából kiindulva:

```
if ( osszeg > vizsgalt )
{
```

```
        document.write( "Az eredmény nagyobb mint " + vizsgalt );
}
else if  (  osszeg  <  vizsgalt  )
{
        document.write( "Az eredmény kisebb mint " + vizsgalt );
}
else
{
        document.write( "Az eredmény pont "  +  vizsgalt );
}
```

Ahogy ebben a példában is látható, az else után elhelyezhetünk egy újabb if-et (else if = különben ha), ezzel egy új feltétel vizsgálatára utasítva a programot, amennyiben az első nem teljesült volna.

Itt fontos megjegyezni, hogy a feltételeket addig vizsgálja a program, míg valamelyik nem teljesül. Tehát ha a legelső összehasonlítás igaz lesz, akkor a többi feltétel már nem kerül vizsgálatra.

Ha egyik feltétel sem teljesülne, akkor az else ág fog lefutni, ha megadtunk ilyet. Amennyiben nem, úgy egyik feltétel alatt elhelyezett programrész sem fog végrehajtásra kerülni.

Természetesen lehetőségünk van az egyes feltételek teljesülésekor futó kódba további feltételeket elhelyezni. Sőt még azokon belül is újabbakat, akár a végtelenségig. Bár mi ilyen bonyolultságú kódot nem fogunk írni, de jó tudni, hogy ezzel a módszerrel már komolyabb programok is készíthetők.

Függvények

A függvények, más szóval funkciók (function) olyan programrészek, amelyek elkülönülnek a programunk fő részétől, viszont bármikor meghívhatók, ha szükségünk lenne rájuk.
Segítségükkel tudunk olyan programrészeket írni, amit a programunk futása során akár többször le kell futtatnunk valamilyen célból.

Nézzük is meg, hogyan tudunk egy egyszerű függvényt írni és lefuttatni azt:

```
function hello()
{
        document.write  (  "Helló világ!"  );
}

hello();
```

Lássuk mit is takar a fenti kód:
A function utasítással adtuk meg, hogy most egy függvény deklarálása fog következni.
Az ezt követő szó (hello) maga a függvény neve, amire a programunkban majd hivatkozni tudunk.
A név után egy üres zárójel- páros jön, melyben olyan változókat helyezhetünk el, amiket szeretnénk a függvényünknek átadni. Erre részletesebben is kitérünk mindjárt.
A { } jelek között pedig azt a programkódot adjuk meg, amelyet a függvény meghívásánál szeretnénk

futtatni.

Magát a függvényt futtatni a függvénynév() ; formában tudjuk.

Fontos tudni, hogy a függvények kódja addig nem fog lefutni, még meg nem hívtuk a függvényt. Mindegy, hogy hol helyezzük el őket a kódban, viszont átláthatósági szempontból célszerű a főprogram elejére gyűjteni őket, vagy külön fájlba.

A fenti példa függvényt lefuttatva kiíratjuk a Helló világ! mondatot.
Ha többször írnánk be egymás után a hello(); függvénymeghívást, akkor annyiszor kapnánk meg ezt a mondatot, ahányszor meghívtuk a függvényt.

Ahogy fentebb említettem, lehetőségünk van értékeket is átadnunk a függvénynek. Alapesetben ugyanis a függvények egy zárt programrészt képviselnek, és nincsenek kapcsolatban a programunk más részeivel. Ezért külön meg kell adnunk, hogy a programunk más részeiből mik azok, amiket megkapjon a függvényünk.

Maradva a kis műveletvégzős példánknál, de már függvényesítve azt:

```
function szamol( szam1, szam2, muvelet )
{
	if ( muvelet == "osszead" )
	{
		var eredmeny = szam1 + szam2;
	}
	else if ( muvelet == "szoroz" )
	{
		var eredemny = szam1 * szam2;
	}
	else
	{
		var eredmeny = "Ismeretlen művelet";
	}

	document.write ( eredmeny );
}


var szam1 = 10;
var szam2 = 25;
var muvelet = "szoroz";

szamol( szam1, szam2, muvelet );
```

Ebben a példában már minden megvan, amit eddig tanultunk: változók, feltételek és egy függvény is.

Lássuk sorban, mit látunk, mi történik, ha lefuttatjuk ezt a kódot:

Elsőnek deklaráltunk egy számol nevű függvényt, melynek megadtunk három paramétert. Két számot és egy műveletet vár, amikor majd meghívjuk.
Mivel alapértelmezésben a függvények nem futnak le önmaguktól, a benne foglalt részt ki is hagyja a program és az azt követő programrészre fog ugrani. Itt deklaráltuk a két számot és műveletnek megadtuk a szorzást.
Majd ezekkel az értékekkel meghívtuk a számol függvényünket.
Fontos tudni, hogy mindig annyi értékkel kell meghívnunk a függvényt, ahány paraméterrel deklaráltuk azt. Ellenkező esetben hibát fog jelezni a programunk és nem fog lefutni.

Tehát meghívtuk a függvényt, átadva neki két számot és egy műveletet. Lássuk, mi történik ekkor a függvényben elhelyezett kódban:
Itt rögtön megvizsgáljuk milyen műveletet adtunk meg. Ebben a példában ez vagy összeadás, vagy szorzás lehet. Ha valami mást adnánk meg, a művelet változó értékeként, akkor az első ág futna le, ahol az eredmény Ismeretlen művelet üzenet lenne. Ellenkező esetben, a két feltétel egyike kerül lefutásra, ahol a kívánt művelettel megkapjuk az eredményt. A feltételek végén pedig kiíratjuk magát az eredmény változó értékét.

Ezzel a példával már jól látszik a függvények, feltételek és változók alkalmazásának lényege. Néhány szám és művelet megadásával, könnyedén tudunk számoltatni bármit anélkül, hogy magát a műveleteket végző programrészt többször megírnánk.

Jó, ha tudjuk, hogy a függvényeket nem csak változó paraméterrel, de közvetlen paraméterrel is meghívhatjuk.
Az előző függvényünket alapul véve, például:

```
szamol( 12, 34, "osszead" );
szamol( 25, 55, "szoroz" );
szamol( 34, 46, "osszead" );
```

Mint látható, három függvény hívással, három műveletet el tudunk végezni, mindössze annyival, hogy a nekik átadott értékeket módosítottuk.

Kapcsolat a HTML elemekkel

Azt már láttuk, hogy a document.write() paranccsal tudunk szöveget, vagy HTML elemeket elhelyezni a weblapunkon. De mi van akkor, ha egy űrlap elemének értékét szeretnénk megkapni, vagy egy űrlap gombjának megnyomásával egy függvényt lefuttatni?
Szerencsére ezek egyike sem túl bonyolult feladat!

Tegyük fel, hogy van egy egyszerű szöveges űrlap mezőnk:

```
<form name="peldaurlap">
      <input type="text" name="szam" />
```

```
</form>
```

Ha ennek a mezőnek az értékét meg szeretnénk kapni egy változóba, akkor azt az alábbi formában tehetjük meg:

```
var szam = document.peldaurlap.szam.value;
```

Tehát deklarálunk egy szam változót, a már ismert módon. Értékeként pedig az űrlap elem elérését kell megadnunk, ahol jelen esetben:

document	- maga a HTML dokumentumunk
peldaurlap	- Az űrlapunk neve
szam	- Ez elemünk neve, melynek értékét szeretnénk megkapni
value	- Végül a value-val (érték) határozzuk meg, hogy a program az elem értékét adja vissza.

Nagyon hasonlóan adhatunk értéket is egy űrlap elemnek:

```
document.peldaurlap.szam.value = 235;
```

Tehát ha egy egyenlőség jellel ellátva egy értéket adunk meg a value után, azt a program beírja az űrlap elemnek, mint érték.

És végül, ha egy függvényünket gombnyomásra szeretnénk lefuttatni:

```
<form name="peldaurlap">
	<input type="button" value="Minta gomb" onclick="fuggveny_1()" />
</form>
```

Egy egyszerű onclick (klikkeléskor) paramétert adunk az input taghez, aminek értékként annak a függvénynek a nevét kell megadnunk, amelyet futtatni szeretnénk megnyomásakor.

Hogy egyben lássuk mindezeket, végezzünk el a korábbi példákban látott műveleteket, immáron űrlap felhasználásával.

A HTML kódunk ehhez a következő:

```
<form name="szamolo">
	<input value="text" name="eredmeny" />

	<input value="text" name="szam1" />
	<input value="text" name="szam2" />

	<input type="radio" name="muvelet" value="osszead"> Összeadás
	<input type="radio" name="muvelet" value="szorzas"> Szorzás

	<input type="button" value="Számol" onclick="szamol()" />
</form>
```

És azt ezt kezelő JavaScript kódunk:

```
function szamol()
{
	var szam1    = document.szamolo.szam1.value;
	var szam2    = document.szamolo.szam2.value;
	var muvelet = document.szamolo.muvelet.value;

	if ( muvelet == "osszead" )
	{
		var eredmeny = szam1 + szam2;
	}
	else if ( muvelet == "szoroz" )
	{
		var eredemny = szam1 * szam2;
	}
	else
	{
		var eredmeny = "Ismeretlen művelet";
	}

	document.szamolo.eredmeny.value = eredmeny;
}
```

A HTML rész nagyon egyszerű. Készítettünk egy űrlapot, két szám beviteli mezővel, egy műveletet választó rádió gombbal, és egy gombbal, mely megnyomáskor meghívja a szamol() JS függvényünket. Valamint van egy eredmény nevű input boxunk is, amelyben majd a kapott eredményt fogjuk megjeleníteni.

A JS rész szinte ugyanaz, mint az előző példákban is használtunk, viszont itt a változókat nem adtuk meg előre, hanem az űrlap megfelelő elemeinek lekérdezésével kaptak értéket.
Miután a program elvégezte a megfelelő műveletet a rádió gombok állapotának megfelelően, azt most nem egy egyszerű document.write()-al fogjuk kiíratni, hanem értékként fogjuk megadni az eredmény nevű beviteli mezőnknek. Ez már tulajdonképpen egy egyszerű számológép alapja.
Hogy minden eddig tanultat egyben lássunk egy kicsit összetettebb példán, írjunk is egy egyszerű számológép programot a következő feladatban.